

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICATION FOR LETTERS PATENT

# Recognizer of Audio-Content in Digital Signals

Inventors:

M. Kivanç Mihçak  
Ramarathnam Venkatesan

ATTORNEY'S DOCKET NO. MS1-645US

1  
2 **TECHNICAL FIELD**

3 This invention generally relates to a technology for improving the  
4 recognition of audio content of digital signals.

5  
6 **BACKGROUND**

7 Digital audio signals offer many advantages over conventional media in  
8 terms of audio quality and ease of transmission. With the ever-increasing  
9 popularity of the Internet, digital audio clips have become a mainstay ingredient of  
10 the Web experience, buoyed by such advances as the increasing speed at which  
11 data is carried over the Internet and improvements in Internet multimedia  
12 technology for playing such audio clips. Everyday, numerous digital audio clips  
13 are added to Web sites around the world.

14 An audio "clip" indicates an audio signal (or bit stream), in whole or part.  
15 A clip may be stored and retrieved, transmitted and received, or the like.

16 As audio clip databases grow, the needs for indexing them and protecting  
17 copyrights in the audio clips are becoming increasingly important. The next  
18 generation of database management software will need to accommodate solutions  
19 for fast and efficient indexing of digital audio clips and protection of copyrights in  
20 those digital audio clips.

21 A hashing technique is one probable solution to the audio clip indexing and  
22 copyright protection problem. Hashing techniques are used in many areas such as  
23 database management, querying, cryptography, and many other fields involving  
24 large amounts of raw data. A hashing technique maps a large block of data (which  
25 may appear to be raw and unstructured) into relatively small and structured set of



1 compilers and message authentication codes (MACs) in cryptographic  
2 applications.

3 Good hashing techniques have long existed for many kinds of digital data.  
4 These functions have good characteristics and are well understood. The idea of a  
5 hashing technique for audio clip database management is very useful and  
6 potentially can be used in identifying audio clips for data retrieval and copyrights  
7 protection.

8 Unfortunately, while there are many good existing functions, digital audio  
9 clips present a unique set of challenges not experienced in other digital data,  
10 primarily due to the unique fact that audio clips are subject to evaluation by human  
11 listeners. A slight pitch or phase shifting of an audio clip does not make much  
12 difference to the human ear, but such changes appear very differently in the digital  
13 domain. Thus, when using conventional hashing functions, a shifted version of an  
14 audio clip generates a very different hash value as compared to that of the original  
15 audio clip, even though the audio clips sound essentially identical (i.e.,  
16 perceptually same).

17 Another example is the deletion of a short block of time from an audio clip.  
18 If the deleted block is short and in an otherwise quiet portion of the clip, most  
19 people will not recognize this deletion in the audio clip itself, yet the digital data is  
20 altered significantly if viewed in the data domain.

21 Human ears are rather tolerant of certain changes in audio clips. For  
22 instance, human ears are less sensitive to changes in some ranges of frequency  
23 components of an audio clip than other ranges of frequency components. Human  
24 ears are also unable to catch small stretching and shrinking of short segments in  
25 audio clips.

1 Many of these characteristics of the human auditory system can be used  
2 advantageously in the delivery and presentation of digital audio clips. For  
3 instance, such characteristics enable compression schemes, like MP3, to compress  
4 audio clips with good results, even though some of the audio clip data may be lost  
5 or go unused. There are many audio clip restoration/enhancement algorithms  
6 available today that are specially tuned to the human auditory system. Commercial  
7 sound editing systems often include such algorithms.

8 At the same time, these characteristics of the human auditory system can be  
9 exploited for illegal or unscrupulous purposes. For example, a pirate may use  
10 advanced audio processing techniques to remove copyright notices or embedded  
11 watermarks from an audio clip without perceptually altering the audio clip. Such  
12 malicious changes to the audio clip are referred to as "attacks", and result in  
13 changes at the data domain.

14 Unfortunately, a human is unable to perceive these changes, allowing the  
15 pirate to successfully distribute unauthorized copies in an unlawful manner.  
16 Traditional hashing techniques are of little help because the original audio clip and  
17 pirated copy hash to very different hash values, even though the audio clips sound  
18 the same.

19 Common Attacks. The standard set of plausible attacks is itemized in the  
20 Request for Proposals (RFP) of IFPI (International Federation of the Phonographic  
21 Industry) and RIAA (Recording Industry Association of America). The RFP  
22 encapsulates the following security requirements:

- 23 • two successive D/A and A/D conversions,
- 24 • data reduction coding techniques such as MP3,
- 25 • adaptive transform coding (ATRAC),

- adaptive subband coding,
- Digital Audio Broadcasting (DAB),
- Dolby AC2 and AC3 systems,
- applying additive or multiplicative noise,
- applying a second Embedded Signal, using the same system, to a single program fragment,
- frequency response distortion corresponding to normal analogue frequency response controls such as bass, mid and treble controls, with maximum variation of 15 dB with respect to the original signal, and
- applying frequency notches with possible frequency hopping.

Accordingly, there is a need for a hashing technique for digital audio clips that allows slight changes to the audio clip which are tolerable or undetectable (i.e., imperceptible) to the human ear, yet do not result in a different hash value. For an audio clip hashing technique to be useful, it should accommodate the characteristics of the human auditory system and withstand various audio signal manipulation processes common to today's digital audio clip processing.

A good audio hashing technique should generate the same unique identifier even though some forms of attacks have been done to the original audio clip, given that the altered audio clip is reasonably similar (i.e., perceptually) to a human listener when comparing with the original audio clip. However, if the modified audio clip is audibly different or the attacks cause irritation to the listeners, the hashing technique should recognize such degree of changes and produce a different hash value from the original audio clip.

## Content Categorization

Like anti-piracy, semantic categorizing of the audio content of audio clips often requires subjective comparisons to other existing audio works. Works of a similar nature are grouped into the same category. The content of audio clips may be semantically classified into any number of categories, such as classical music, conversation, hard rock, easy listening, polka, lecture, country, and the other such semantic categories.

Typically, such semantic categorization is subjectively determined by manual (i.e., human) subjective analysis of a work so that it may be grouped with an existing category. No such technique exists for automatically (i.e., without substantial human involvement) analyzing and categorizing the semantic audio content of audio clips.

## SUMMARY

Described herein is a technology for recognizing the audio content of digital signals. The technology determines one or more hash values for the original audio content of a digital signal.

In one implementation described herein, audio content recognition facilitates identification of the original audio content and other audio content that is substantially perceptually same as the original audio content. An identification hash value uniquely identifies the original audio content to determine if the content of another digital signal is substantially perceptually same.

In another implementation described herein, audio content recognition facilitates semantic categorization of such original content so that it may be

1 grouped with other audio works in the same semantic category. A categorization  
2 hash values is used to group the original content with other audio works having  
3 similar categorization hash values; thereby, semantically classifying audio works  
4 having similar hash values as a semantic category of similar audio works.

5 This summary itself is not intended to limit the scope of this patent. For a  
6 better understanding of the present invention, please see the following detailed  
7 description and appending claims, taken in conjunction with the accompanying  
8 drawings. The scope of the present invention is pointed out in the appending  
9 claims.

## 10 **BRIEF DESCRIPTION OF THE DRAWINGS**

11  
12 The same numbers are used throughout the drawings to reference like  
13 elements and features.

14 Fig. 1 is a schematic block diagram showing an embodiment of an  
15 implementation of the present invention claimed herein.

16 Figs. 2A-2C illustrate an implementation of the MCLT transform in  
17 accordance with an implementation of the present invention claimed herein.

18 Figs. 3A-1 and 3A-1 illustrate the time-frequency representation and the  
19 significance map, respectively, of given audio clip.

20 Figs. 3B-1 and 3B-1 illustrate the time-frequency representation and the  
21 significance map, respectively, of another audio clip.

22 Fig. 4 is a flow diagram showing an illustrative methodological  
23 implementation of an implementation of the present invention claimed herein.

24 Figs. 5A-5C illustrate some of the tasks performed by a statistic estimator  
25 in accordance with an implementation of the present invention claimed herein.



Fig. 6 is a flow diagram showing an illustrative methodological implementation of an implementation of the present invention claimed herein.

Fig. 7 is a flow diagram showing an illustrative methodological implementation of an implementation of the present invention claimed herein.

Fig. 8 is a flow diagram showing an illustrative methodological implementation of an implementation of the present invention claimed herein.

Fig. 9 is an example of a computing operating environment capable of implementing an implementation of the present invention claimed herein.

## **DETAILED DESCRIPTION**

The following description sets forth one or more specific embodiments of an recognizer of audio-content in digital signals that incorporate elements recited in the appended claims. These embodiments are described with specificity in order to meet statutory written description, enablement, and best-mode requirements. However, the description itself is not intended to limit the scope of this patent.

Described herein are one or more exemplary implementations of a method and system of fusing portions of a print medium. The inventors intend these exemplary implementations to be examples. The inventors do not intend these exemplary implementations to limit the scope of the claimed present invention. Rather, the inventors have contemplated that the claimed present invention might also be embodied and implemented in other ways, in conjunction with other present or future technologies.

An exemplary embodiment of an recognizer of audio-content in digital signals may be referred to as an “exemplary audio recognizer.”

## Incorporation by Reference

The following co-pending patent applications are incorporated by reference herein (which are all assigned to the Microsoft Corporation):

- U.S. Patent Application Serial No. 09/390271, entitled “A Technique for Watermarking an Image and a Resulting Watermarked Image” filed Sept. 7, 1999;
- U.S. Patent Application Serial No. 09/390272, entitled “A Technique for Detecting a Watermark in a Marked Image” filed on Sept. 7, 1999;
- U.S. Patent Application Serial No. 09/316,899, entitled “Audio Watermarking with Dual Watermarks” filed on May 22, 1999;
- U.S. Patent Application Serial No. 09/614,660, entitled “Improved Stealthy Audio Watermarking” filed on July 12, 2000;
- U.S. Patent Application Serial No. \_\_\_\_\_, entitled “Robust Recognizer of Perceptually Similar Content” filed on April 24, 2001;
- U.S. Patent Application Serial No. \_\_\_\_\_, entitled “Derivation and Quantization of Robust Non-Local Characteristics for Blind Watermarking” filed on April 24, 2001;
- U.S. Patent Application Serial No. 09/259,669, entitled “A System and Method for Producing Modulated Complex Lapped Transforms” filed on February 26, 1999; and
- U.S. Patent Application Serial No. 09/421,986, entitled “System and Method for Hashing Digital Images” filed on October 19, 1999.

1 The following U.S. Patent is incorporated by reference herein: U.S. Patent  
2 No. 6,029,126, entitled "Scalable Audio Coder and Decoder" issued on February  
3 22, 2000, and assigned to the Microsoft Corporation.

#### 4 **Introduction**

5  
6 Implementations, described herein, of the exemplary audio recognizer may  
7 be implemented (whole or in part) by an audio-content recognition system 100  
8 and/or by a computing environment like that shown in Fig. 9.

9 An exemplary implementation is described herein as a technique for  
10 generally recognizing audio content of digital audio signals by hashing such  
11 signals to generate one or more hash values for each signal.

12 An exemplary implementation is described herein as a technique for  
13 identifying audio content of such signals by comparing identification hash values  
14 of signals. This exemplary implementation generates the same unique identifier  
15 (e.g., hash value) even though some forms of attacks have been done to the  
16 original digital audio signal, given that the altered signal is perceptually same to a  
17 human listener when comparing the altered signal with the original signal.  
18 However, if the altered signal is perceptually audibly different or the attacks cause  
19 irritation to the listeners, the hashing technique recognizes such degree of changes  
20 and produces a different hash value from the original signal.

21 An exemplary implementation is described herein as a technique for  
22 categorizing audio content of such signals by comparing categorization hash  
23 values of signals.

1 An exemplary implementation described herein is a technique that may be  
2 combined with techniques described in documents which are incorporated herein  
3 by reference.

#### 4 **Exemplary Applications**

6 Implementations, described herein, of the exemplary audio recognizer are  
7 suitable for numerous applications, including the following (which are provided as  
8 examples and not as limitations): identification, searching & sorting in a database,  
9 semantic content categorization, and anti-piracy applications (such as  
10 watermarking). Some of the exemplary applications are discussed below:

11 Locating content in a database. Identification hash values may be stored  
12 and associated with specific audio content of a digital audio signal. When  
13 searching for such content, a search engine may look for a given hash value to  
14 locate the content. This is much more efficient than conventional techniques for  
15 searching for audio content in a database.

16 Semantic content categorization. This includes approximate matching of  
17 content between two digital audio signals. The hashing techniques of the  
18 exemplary embodiments have an intermediate hash value, which can be used to  
19 compare if two given items are similar. This hash value may also be called a  
20 categorization hash value.

21 This categorization hash value may be used to semantically classify audio  
22 content of audio works. Works of a similar nature tend to have categorization  
23 hash values that cluster together. Thus, these values are proximally near each  
24 other. This proximal range may be subjectively determined. For some semantic  
25 categories, the range may be large and for others it may be small.

1 The categorization hash can be computed incrementally. This means that if  
2 the exemplary embodiment may slide the window of the audio clip, one may  
3 compute the hash value on the new window from the old window without doing  
4 substantial reworking on the part that is common to old and new windows.

5 Anti-piracy search efforts. One may search for the pirated content of audio  
6 signals on the web, which might have been subjected to watermarking and/or  
7 malicious attacks, using a web crawler and a database of signal hash values.

8 Content dependent key generation (for watermarks): Since the  
9 watermarking technique must use secret keys, using the same key for millions of  
10 pieces of content may compromise the key. Thus, an exemplary embodiment may  
11 generate a hash value for an audio signal that may function as signal dependent  
12 keys. For an attacker without the knowledge of the secret key used, the hash value  
13 of a given content will be unpredictable.

## 14 Hashing

15  
16 The use of hashing techniques, which map long inputs into short random-  
17 looking (i.e., uniformly distributed) outputs, are many and indeed wide-ranging:  
18 compilers, checksums, searching and sorting techniques, cryptographic message  
19 authentication, one-way hashing techniques for digital signatures, time stamping,  
20 etc. These techniques usually accept binary strings as inputs and produce a fixed  
21 length ("L") hash value. These techniques use some form of random seeds (i.e.,  
22 keys).

23 The hash values produced by such techniques are viewed as useful because  
24 they typically have following desirable characteristics:  
25

- Almost Uniformly Distributed—For any given input, the output hash value are uniformly distributed among the possible L-bit outputs.
- Approximate Pairwise Independent—For two distinct inputs, the corresponding outputs are statistically almost independent of each other.

The hashing technique implemented by various systems and methods, described herein, is denoted as  $H$ . Given an input signal  $I$ , the hashing technique  $H$  produces a short binary string  $X$ , as follows:

$$H(I) = X$$

The hashing technique  $H$  has the following properties:

- For any signal  $I_i$ , the hash of the signal,  $H(I_i)$ , is approximately uniformly distributed among binary strings of equal length.
- For two distinct signals,  $I_1$  and  $I_2$ , the hash value of the first signal,  $H(I_1)$ , is approximately independent of the hash value of the second signal,  $H(I_2)$ , in that given  $H(I_1)$ , one cannot predict  $H(I_2)$ .
- If two signals  $I_1$  and  $I_2$  are perceptually the same or similar, the hash value of the first signal,  $H(I_1)$ , should equal the hash value of the second signal,  $H(I_2)$ .

1 The hashing techniques of the implementations, described herein, are  
2 similar in many respects to the hashing techniques described in some of the  
3 documents which are incorporated herein by reference. The hashing techniques,  
4 described herein, are particularly tailored to accommodate characteristics of the  
5 human auditory system and withstand various audio signal manipulation processes  
6 common to today's digital audio signal processing.

### 7 8 Perceptually Same and Perceptually Distinct

9 The exemplary audio recognizer treats two "perceptually same" audio  
10 signals as the same. Herein, a pair of digital audio signals are "perceptually same"  
11 when their identification hash values are the same (alternatively, substantially the  
12 same).

13 "Perceptually same" audio signals include those that sound as if they are  
14 they are substantially same to the human ear. As a first step in determining what it  
15 means to be perceptually same, one can use the standard Turing test used for  
16 formulating, for example, pseudo-randomness in cryptography. A listener is  
17 played two audio clips, one after another. Then the clips are played in random  
18 order and the listener should match up the clips. If the listener has no better than  
19 roughly a fifty-percent chance, the clips can be deemed perceptually same.

20 However, the use of the term is more general than that defined by the  
21 Turing test. Clips that have passed the Turing test may still be considered  
22 perceptually same when the listener is able to distinguish them only based on  
23 "slight and insubstantial differences." Regardless of such slight and insubstantial  
24 differences, the listener can clearly state that these clips "are the same audio clips  
25 for all practical purposes."

1  
2 In contrast, a “perceptually distinct” digital goods is generally the converse  
3 of “perceptually same” digital goods. This may also be called “perceptually  
4 different” or “perceptually distinguishable”.

### 5 6 Perceptually Similar

7 The exemplary audio recognizer treats two “perceptually similar” audio  
8 signals as different signals that should be categorized as similar. Herein, a pair of  
9 digital audio signals are “perceptually similar” when their categorization hash  
10 values are close in value (i.e., proximal). Signals that are “perceptually similar”  
11 may also be “perceptually same” if their identification hash values are the same.

### 12 13 Exemplary Hashing Technique of the Exemplary Implementations

14 The hashing technique of the exemplary implementations is an irreversible  
15 hashing function that operates on audio clips and yields a final hash value of a  
16 binary string of specified length. It also yields one or more intermediate has  
17 values. The hashing techniques of the exemplary implementations have the  
18 following criteria:

- 19 • The hash values are almost uniformly distributed with high  
20 probability.
- 21 • With high probability, the hash values for “perceptually audibly  
22 distinct” audio clips are different.
- 23 • With high probability, the hash values for “perceptually audibly  
24 same” audio clips are the same.



Assume  $X$  denotes a particular audio clip,  $X'$  denotes a modified version of this clip which is “perceptually same” as  $X$ , and  $Y$  denotes a “perceptually distinct” audio clip. Assume  $L$  is the final length of the hash and  $H(\cdot)$  represents a hashing function. The following is a performance measure on the hash values that are produced:

$$D(H(X), H(Y)) = \frac{\sum_{i=1}^L H(X_i) \otimes H(Y_i)}{L} \quad (1)$$

where  $H(X_i)$  and  $H(Y_i)$  are the values of  $H(X)$  and  $H(Y)$  respectively at the  $i^{\text{th}}$  location, and  $\otimes$  stands for the  $XOR$  operation. Formula 1 is the ratio of the number of locations where  $H(X)$  and  $H(Y)$  are different to the total length of the hash.

The following are examples of a family of hashing functions, for use with the exemplary embodiments, with parameters:  $p$ ,  $L$  (where  $p \approx 1/2^L$ ).

- Randomization:

$$\Pr[H(X) = \alpha] = p \approx \frac{1}{2^L}, \forall \alpha \in \{0,1\}^L,$$

- Perceptually distinct audio clips  $X$ ,  $Y$ :

$$\Pr[H(X) = \alpha \mid H(Y) = \beta] \approx \Pr[H(X) = \alpha], \forall \alpha, \beta \in \{0,1\}^L,$$

Thus,  $H(X)$  is not equal to  $H(Y)$ , with overwhelming probability

- Perceptually same audio clips:

$$\Pr[H(X) = H(X')] \approx 1.$$

Therefore, apart from the randomization issue, the difference between identification hash values of audio clips may be represented as follows:

$$D(H(X), H(X')) = 0, D(H(X), H(Y)) > 0, \quad (2)$$

for all possible different (“perceptually distinct”) audio clips X, Y and for all possible “perceptually same” audio clips X, X’.

Intermediate hash value. In the exemplary embodiments described herein, an intermediate hash value is calculated before the final (i.e., identification) hash value is calculated. The intermediate hash values are of length M, where  $M > L$  and have the following separation property:

$$D(H(X), H(X')) < 0.2, D(H(X), H(Y)) > 0.40, \quad (3)$$

In the exemplary embodiments, M has a value within range of 5L to 10L. To determine the intermediate hash value, decoding stages of first order Reed-Müller codes employed with a specialized pseudo-norm. Given the intermediate hash, the exemplary embodiment uses some generic techniques (e.g., list-decoding procedures) to generate a binary string with desired properties.

For more information on generating intermediate hash values, see U.S. Patent Application Serial No. \_\_\_\_\_, entitled “Robust Recognizer of Perceptually Similar Content” filed on April \_\_, 2001.

## Exemplary Audio Recognizer

Fig. 1 shows the audio-content recognition system 100, which is an example of an embodiment of the exemplary audio recognizer. The system 100 includes a transformer 110, a statistics estimator 120, an adaptive quantizer 130, and an error-correction decoder 140.

The transformer 110 obtains a digital audio signal 105 (such as an audio clip). It may obtain the signal from nearly any source, such as a storage device or over a network communications link. The transformer 110 puts the signal 105 in canonical form using a set of transformations. Specifically, the exemplary audio recognizer employs MCLT (Modulated Complex Lapped Transform) and obtain time-varying spectral characteristics,  $T_x$  112, of the audio clip.

The statistics estimator 120 applies a randomized interval transformation in order to extract audible statistics,  $\mu_x$  122, of the signal. These audible statistics are expected to represent the audio signal in an irreversible manner while introducing robustness against attacks. For perceptually same audio clips, these statistics are likely to have close values (under suitable notions of metric) whereas for perceptually distinct audio clips they are far apart.

The adaptive quantizer 130 applies randomized rounding (i.e., quantization) to the outputs of the statistics estimator 120. The adaptive quantizer produces outputs,  $q_x$  132, that are represented as binary vectors. Alternatively, the quantizer 130 may be non-adaptive.

The error-correction decoder 140 uses the decoding stages of an error correcting code to map similar values to the same point. The final output,  $h_x$  142, is produced by the decoder 140. This final output is the final hash value. The

1 decoder also produces an intermediate hash value. Alternatively, error-correction  
2 decoder 140 may be a randomized Vector Quantizer (VQ) or list-decoder, or  
3 other similar devices.

4 The decoder 140 produces intermediate hash values such that the  
5 normalized Hamming distance between intermediate hash values of perceptually  
6 distinct audio signals is greater than 0.40 and the normalized Hamming distance  
7 between intermediate hash values of perceptually similar audio signals is less than  
8 0.20. Of course, these ranges are provided as examples only and not for  
9 limitation.

10 Each of aforementioned components of the audio-content recognition  
11 system 100 of Fig. 1 is explained in greater detail below.

#### 12 The Transformer 110 and MCLT

13 MCLT is a complex extension of MLT (Modulated Lapped Transform).  
14 MLT was introduced in and is used in many audio-processing applications, such  
15 as DolbyAC-3, MPEG-2. MCLT basis functions are found in pairs to produce real  
16 and complex parts separately. These basis functions are derived from MLT and  
17 they are phase-shifted versions of each other. It can be shown that they possess  
18 some intuitively pleasing properties such as perfect reconstruction and  
19 approximate shift invariance. The MCLT is a special case of 2x oversampled DFT  
20 (Discrete Fourier Transform) filter bank.  
21

22 The MCLT is discussed more fully in U.S. Patent Application Serial No.  
23 09/259,669, entitled "A System and Method for Producing Modulated Complex  
24 Lapped Transforms," which is incorporated by reference herein.  
25

Figs. 2A-2C illustrate the MCLT scheme, like the one employed by the transformer 110 of the audio-content recognition system 100 of Fig. 1. Fig. 2A shows a timeline 210 of an input sequence of an audio signal (which is not shown). The sequence is broken into overlapping “blocks” (blocks 212-218) along the timeline, so that neighboring blocks intersect by half.

As shown in Fig. 2B, the MCLT transform is applied, by MCLT transformer 230, applied independently to each block (such as block “i”) to produce spectral decomposition of size M. Assuming that the analysis and synthesis filters are of length 2M, then the number of the frequency bands is M in the spectral domain.

Fig. 2C represents the combination of the spectral decomposition of the blocks in order to form the time-frequency decomposition,  $T_x$ . The MCLT transform of the blocks are combined into a matrix to obtain the time-frequency representation of an audio clip. For Fig. 2C, Let M be the number of frequency bands and N be the number of the blocks and  $T_x$  is the MCLT matrix and  $T_x(i, j)$  is the MCLT value at location (i, j) where  $j = 0, 1, \dots, N-1$ .

MCLT can be used to define a “hearing threshold matrix”  $H_x$ , which is the same size as  $T_x$ , such that if  $T_x(i, j) \geq H_x(i, j)$ , then  $T_x(i, j)$  is audible.

#### Randomized Interval Transformation (Statistics Estimation)

The following is the definition of the significance map  $S_x$ :  $S_x(i, j) = 1$ , if  $T_x(i, j) \geq H_x(i, j)$ ; otherwise  $S_x(i, j) = 0$ ; where  $i = 0, 1, \dots, M-1$  and  $j = 0, 1, \dots, N-1$ .

Figs. 3A-1 and 3A-2 illustrate the time-frequency representation and corresponding significance map for an exemplary audio clip A. Figs. 3B-1 and

1 3B-2 illustrate the time-frequency representation and corresponding significance  
2 map for a different exemplary audio clip, clip B. Only the low-frequency portions  
3 of the time-frequency representations and the significance maps are depicted in  
4 these figures for the purposes of convenience.

5 As shown in Figs. 3A-1 and 3B-1, there is a striking pattern in time-  
6 frequency representation of the audio clips. Furthermore, this pattern has a slowly  
7 varying structure—with respect to both time and frequency. The exemplary audio  
8 recognizer captures this existing structure in a compact fashion via randomized  
9 interval transformations (i.e., “statistics estimation”).

10 The statistics estimator 120 estimates statistics reflect characteristics of the  
11 signal 105. In order to achieve this purpose, the estimator 120 carries out  
12 estimation of statistics in the time-frequency plane. The estimator 120 exploits  
13 both local and global correlations. Correlations exist both along frequency axis  
14 and along the time axis. The estimator 120 may, for example, employ one or two  
15 exemplary approaches. Approach I operates along the frequency axis for each  
16 block (i.e., exploits correlations along the frequency axis for a given block).  
17 Approach II operates along the time axis for each frequency subband; and  
18 therefore, it exploits correlations along time.

### 19 **Methodological Implementations of Approach I and Approach II:**

20  
21 Fig. 4 show methodological implementations of both Approach I and  
22 Approach II, which is performed by the estimator 120 of the audio-content  
23 recognition system 100 (or some portion thereof). These methodological  
24 implementations may be performed in software, hardware, or a combination  
25 thereof.

1 The methodology of Approach II is very similar to Approach I. The main  
2 difference is that in Approach I correlations along frequency are exploited instead  
3 of correlation along time while in Approach II correlations along time are  
4 exploited instead of correlation along frequency.

5 At 410 of Fig. 4, for each block, the estimator 120 determines if there exist  
6 sufficiently many entrees exceeding the hearing thresholds. If not pass to the next  
7 block, else collect the “significant” coefficients into a vector of size  $M' \leq M$ .  
8 More generally, the columns of  $T_x$  are used in Approach I and the rows of  $T_x$  are  
9 used in Approach II.

10 At 412, the estimator 120 performs a randomized interval transformation.  
11 Fig. 5A illustrates the concept of randomized “splitting” at a single level. At a  
12 single level of randomized splitting of a vector 510 (in either the frequency or time  
13 domain depending upon the Approach), first a randomization interval 520 (i.e.,  
14 randomization region) is found that is to be symmetric around the midpoint 522.  
15 The ratio of the length of the randomization interval to the length of the whole  
16 vector 510 is the randomization tuning parameter for splitting. This may be a user-  
17 specified value.

18 Fig 5A shows that a random point 524 is chosen within this interval 520.  
19 This random point 524 is the point to do the splitting; as a result, two “chunks” are  
20 formed. Then this procedure is carried out a specified number of times—each time  
21 is a “level” of the splitting. The level is a function of  $M'$  and the expected number  
22 of coefficients desired within the smallest chunk. Hence, the expected number of  
23 coefficients that are within the smallest chunk is a user-specified parameter and  
24 determines the “level” for each block. As an example, in Fig. 5B, the recursive  
25 splitting is carried out twice, hence level = 2.

At 414 of Fig. 4, the first order statistics at each level for each chunk are estimated once the split points are randomly found. As an example, consider Fig. 5B. At second level of splitting, there are a total of four chunks (chunks 541, 542, 543, 544), and the estimator 120 collects the arithmetic means of these four chunks (alternatively, other statistical values can be collected, such as variance). Then the estimator 120 proceeds to the first level of splitting, and the estimator 120 computes the arithmetic means at that level—namely, arithmetic means of chunks 545 and 456 in Fig. 5B). Finally, at the zeroth level, the estimator 120 performs the arithmetic mean computation for the whole vector 510 (namely, chunk 547). All of these arithmetic means are collected in a statistics vector.

At 416, the process returns back to the beginning of the loop at 405 and repeat the steps of blocks 410-416 until all blocks have be processed. After all blocks have been processed, the loop ends at 418.

Fig. 5C illustrates the difference of Approach I 560 and Approach II 570 on the time-frequency representation 550 of an audio signal.

In Approach I, the estimator 120 estimates the statistics for each time block. With Approach I, the result of the estimator 120 is a statistics vector  $\mu_f$  to denote the estimated means along the *frequency* axis.

In Approach II, the estimator 120 estimates the statistics for each frequency subband. With Approach II, the result of the estimator 120 is a statistics vector  $\mu_t$  to denote the estimated means along the *time* axis.

Although both Approaches I and II of the exemplary embodiment are designed to estimate the first order statistics at this point, an alternative embodiment may include an estimation of any order statistics. In tests, it has been observed that first order statistics yield better results than second order statistics at



1 this point. Regardless of which approach is employed (Approach I or II), the  
2 function of the adaptive quantizer 130 and the error correction decoder 140 is the  
3 same.

4 Another exemplary embodiment employs a third approach, Approach III,  
5 which combines Approaches I and II. In this approach, statistics are estimated  
6 based on random rectangles in the time-frequency plane. The shape of these  
7 rectangles may be adjusted to capture the appropriate characteristics.

### 8 9 Input Adaptive Quantization

10 The adaptive quantizer 130 produces discrete level of outputs given the  
11 statistics vector as the input. Meanwhile, the adaptive quantizer 130 effectively  
12 increases robustness properties of the hashing techniques (employed by the  
13 exemplary audio recognizer) and augments the amount of randomization.

14 In signal processing, the traditional way of producing discrete level of  
15 outputs from continuous level of inputs is termed as “quantization.” Assume  $Q$  is  
16 the number of quantization levels (i.e., the cardinality of the set of discrete level of  
17 outputs). Also assume  $\mu(j)$  denote the  $j^{\text{th}}$  element of a given statistics vector  $\mu$  and  
18  $\mu'(j)$  denote its quantized version. In conventional quantization schemes, the  
19 quantization rule is completely deterministic and given by

$$\Delta_i \leq \mu(j) < \Delta_{i+1} \Rightarrow \mu'(j) = i, i = 0, 1, \dots, Q-1,$$

22  
23 where the interval  $[\Delta_i, \Delta_{i+1})$  is termed as  $i^{\text{th}}$  quantization bin. With the description  
24 herein of the exemplary embodiment, the focus is on the location of quantization  
25 bins rather than the reconstruction levels. In traditional quantization schemes in

1 signal processing, the reconstruction levels are significant since quantization is  
 2 usually applied as a part of a compression scheme. However, apart from the  
 3 indexing issues, the reconstruction levels are not of significance for the exemplary  
 4 embodiment described herein. Therefore, without loss of generality, the  
 5 reconstruction levels are assumed to be given by integer indexes from the set  
 6  $(0,1,...,Q-1)$ .

7 Typically, the input statistics vector comes from a distribution that is highly  
 8 biased at some points. To account for this “colored” nature of the statistics  
 9 distribution, the exemplary embodiment employs an “adaptive quantization”  
 10 scheme, which takes of possible arbitrary biases at different locations of the  
 11 distribution of the statistics. In particular, the exemplary embodiment uses a  
 12 normalized histogram of  $\mu$  as the distribution of the statistics. A normalized  
 13 histogram is usually very resistant against “slightly inaudible” attacks, thus such  
 14 an adaptive scheme does not really bring a burden in terms of robustness.  
 15 Furthermore, a normalized histogram approximates the p.d.f. (probability density  
 16 function) of the marginal density functions of  $\mu(j)$  are the same for all  $j$ , and the  
 17 approximation error diminishes as the size of  $\mu$  tends to infinity. Thus, within  
 18 the exemplary embodiment  $\{\Delta_i\}$  is designed such that

$$\int_{\Delta_{i-1}}^{\Delta_i} p_{\mu}(t)dt = \frac{1}{Q}, i = 0,1,...,Q-1.$$

22 where  $p_{\mu}$  stands for the normalized histogram of the input statistic vector  $\mu$ . The  
 23 intervals  $[\Delta_{i-1}, \Delta_i)$  determine the quantization bins. Furthermore, define the  
 24 “central points”,  $\{C_i\}$ , such that

$$\int_{\Delta_{i-1}}^{C_i} p_{\mu}(t)dt = \int_{C_i}^{\Delta_i} p_{\mu}(t)dt = \frac{1}{2Q}, i = 0,1,...,Q-1.$$

Now around each  $\Delta_i$ , we introduce a randomization interval  $[L_i, U_i]$  such that

$$\int_{L_i}^{\Delta_i} p_{\mu}(t)dt = \int_{\Delta_i}^{U_i} p_{\mu}(t)dt, i = 0, 1, \dots, Q-1,$$

In other words, the randomization interval is symmetric around  $\Delta_i$  for all  $i$  and we also impose the constraint that  $C_i \leq L_i$  and  $U_i \leq C_{i+1}$ . The exact location of these randomization intervals are determined by “Randomization Factor”,

$$\text{Randomization Factor} = \frac{\int_{L_i}^{\Delta_i} p_{\mu}(t)dt}{\int_{\Delta_{i-1}}^{\Delta_i} p_{\mu}(t)dt}, i = 0, 1, \dots, Q-1$$

where “randomization factor” is a parameter that determines the amount of randomization at the output of quantization and is a user-specified number that can clearly take values in the range  $[0, 1/2]$ . This the p.d.f.-adaptive randomized quantization rule of the exemplary embodiment:

$$L_i \leq \mu(j) \leq U_i \Rightarrow \mu'(j) = \begin{cases} i & \text{with probability } \frac{\int_{L_i}^{\mu(j)} p_{\mu}(t)dt}{\int_{L_i}^{U_i} p_{\mu}(t)dt} \\ i-1 & \text{with probability } \frac{\int_{\mu(j)}^{U_i} p_{\mu}(t)dt}{\int_{L_i}^{U_i} p_{\mu}(t)dt} \end{cases}$$

and

$$C_i \leq \mu(j) \leq L_i \Rightarrow \mu'(j) = i-1 \text{ with probability } 1,$$

$$U_i \leq \mu(j) < C_{i+1} \Rightarrow \mu'(j) = i \text{ with probability } 1$$

1  
2 Under such a randomized quantization rule, if  $L_i \leq \mu(j) \leq U_i$ , then  $E[\mu'(j)] =$   
3  $\Delta_i$ . This choice of the “Randomization Factor” offers a trade-off: As this factor  
4 increases, the amount of randomization at the output increases, which is a desired  
5 property; however, this also increases the chances of being vulnerable to attacks  
6 and modifications especially because in that case the security of the system relies,  
7 to a great extent, on keeping the randomization key as a secret. Thus, choosing a  
8 suitable range for “Randomization Factor” is a delicate issue. In the exemplary  
9 embodiment, a user-specified input parameter determines this issue. With such  
10 user-specified input parameter, the user may balance the degree of randomization  
11 desired against security concerns to achieve a customized result.

#### 12 Error Correction Decoding

13  
14 Once the exemplary embodiment has the quantized statistics, the next step  
15 is to convert these values into a binary bit stream (i.e., digital signal) and shorten  
16 the length of the overall stream in such a way that “perceptually same” audio clips  
17 are mapped to binary strings that are close to each other and “perceptually  
18 distinct” audio clips are mapped to binary strings that are far away from each  
19 other.

20 In order to achieve this purpose, the exemplary embodiment employs first  
21 order Reed-Müller codes at this point. Those who are skilled in the art understand  
22 and appreciate that other error correction schemes as well as possibly randomized  
23 Vector Quantization techniques may be employed and remain within the spirit and  
24 scope of the claimed invention.  
25

Reed-Müller codes are a class of linear codes over GF(2) that are easy to describe and have an elegant structure. The generator matrix  $G$  for the first order Reed-Müller code of blocklength  $2^m$  is defined as an array of blocks:

$$G = \begin{bmatrix} G_0 \\ G_1 \end{bmatrix}$$

where  $G_0$  is a single row consisting of all ones and  $G_1$  is a matrix of size  $m$  by  $2^m$ .  $G_1$  is formed in such a way that each binary  $m$ -tuple appears once as a column. Thus, the resulting generator matrix is of size  $m+1$  by  $2^m$ . More details on error correcting codes and Reed-Müller codes by found in *Theory and Practice of Error Control Codes* (1983) by R. Blahut.

Although there exist computationally efficient techniques for decoding with Reed-Müller codes (decoding with majority logic), the exemplary embodiment is using an exhaustive search on the input word space for the sake of simplicity. Unlike traditional decoding schemes that use Hamming distance as the error metric, the decoding scheme of the exemplary embodiment uses an error measure, which is termed as “Exponential Pseudo Norm” (EPN). It is more suitable than traditional error metrics (such as Hamming distance) for multimedia (image and audio) hashing problems.

Assume  $\underline{x}_D$  and  $\underline{y}_D$  are two vectors of length  $L$  such that each component of these vectors belongs to the set  $\{0,1,\dots,Q-1\}$  where  $\log_2 Q$  is a positive integer. Similarly, assume  $\underline{x}$  and  $\underline{y}$  are binary representations of the vectors  $\underline{x}_D$  and  $\underline{y}_D$  respectively, where each decimal component is converted to the binary format by using  $\log_2 Q$  bits. The lengths of  $\underline{x}$  and  $\underline{y}$  are therefore going to be both  $L \log_2 Q$ . EPN is defined between the binary vectors  $\underline{x}$  and  $\underline{y}$  as

$$EPN(x, y) = \sum_{i=1}^{\Delta} K^{|x_D(i) - y_D(i)|},$$

where  $x_D(i)$  and  $y_D(i)$  denote the  $i$ th elements of the vectors  $\underline{x}_D$  and  $\underline{y}_D$  respectively.  $EPN(\underline{x}, \underline{y})$  is actually a function of  $Q$  and  $K$  as well; however, for the sake of having a clean notation the exemplary embodiment are embedding these values in the expression and simply assuming that these values are known within the context of the problem.

$Q$  is the number of quantization levels, and  $K$  is the “exponential constant” that determines how  $EPN$  penalizes large distances more. The results are approximately insensitive to the value of  $K$  if it is chosen to be large enough. Since part of the aim of the exemplary embodiment is to clearly distinguish between close and far values in the decimal representations of binary strings,  $EPN$  is suitable for incorporations into the hashing techniques of the exemplary embodiment.

Examples of methodological acts performed by the error correction decoder 140 are as follows:

- Divide the quantized data into chunks of length that is user-specified.
- Convert them into binary format by using  $\log_2 Q$  bits for each component, where  $Q$  is the number of quantization levels.
- Form the generator matrix of first order Reed-Müller code where the length of the codewords is as close as possible to the length of the binary representation of the chunks.

- For each possible input word (there are a total of  $2^{m+1}$  possible input words for a generator matrix of size  $m+1$  by  $2^m$ ), generate the corresponding output word.
- Find the EPN between each corresponding output word and the quantized data.
- Pick up the input word that yields the minimum amount of EPN.

### **Methodological Implementation of the Exemplary Audio-Content Recognizer**

Fig. 4 shows an illustrative methodological implementation of the exemplary audio-content recognizer performed (wholly in part) by the audio-content recognition system 100 (or some portion thereof). This methodological implementation may be performed in software, hardware, or a combination thereof.

### **Audio-Content Identification Methodological Implementation**

Fig. 6 illustrates an audio-content identification methodological implementation of the exemplary audio-content recognizer. At 610 of Fig. 6, the exemplary audio-content recognizer retrieves a subject audio clip from a database of audio clips or some other source of such audio signals. Once a subject clip is chosen, the exemplary audio-content recognizer, at 612, transforms it, in accordance with the transformation performed by the transformer 110, described above.

At 614 of Fig. 6, the exemplary audio-content recognizer estimates statistics of the transformed clip, in accordance with the estimation performed by

1 the statistics estimator 120, described above. At 616, the exemplary audio-content  
2 recognizer adaptively quantizes the estimated statistics of the transformed clip, in  
3 accordance with the quantization performed by the adaptive quantizer 130,  
4 described above. At 618, the exemplary audio-content recognizer performs error-  
5 correction decoding on the results of the adaptive quantization, in accordance with  
6 the decoding performed by the decoder 140, described above.

7 At 620 of Fig. 6, it determines the hash values based upon the result of the  
8 above steps. The hash values include an intermediate hash value (i.e.,  
9 categorization hash value) and a final hash value. [ These hash values are  
10 recognition representations of the audio content of the original audio clip. That is  
11 because these hash values may be used to recognize (and even identify) the audio  
12 content within an audio clip.

13 At 622 of Fig. 6, the resulting hash values are displayed and stored. These  
14 values are stored in a database in association with the original subject clip from  
15 which the values were calculated.

### 16 Piracy Detection Methodological Implementation

17  
18 Fig. 7 illustrates a piracy detection methodological implementation of the  
19 exemplary audio-content recognizer. At 756 of Fig. 7, the exemplary audio-  
20 content recognizer retrieves a hash value of a selected audio clip. More  
21 particularly, it retrieves the final hash value (i.e., identification hash value) of such  
22 clip from a database of audio clips or some other source of such clips.

23 Fig. 7 also shows the audio-content identification method of Fig. 6 at block  
24 752. The method 752 calculates a final hash value of an audio clip 750 that is  
25 suspected of being a copy of the selected clip retrieved by block 756. At 754, the



1 exemplary audio-content recognizer retrieves the calculated final hash value of the  
2 suspect audio clip 750 from the audio-content identification method of block 752.  
3 Of course, this can be reversed so that the method 752 provides the hash value of  
4 the *selected* clip while block 752 provides the hash value of the *suspected* clip.

5 At 758, the exemplary audio-content recognizer compares the hash values  
6 of the two clips (suspect clip 750 and selected clip of 756) to determine if they  
7 substantially match. Substantially matching means that the two hash values are  
8 close enough in value to reasonably conclude that the two clips have the same  
9 hash values within a margin of error.

10 If the result of such comparison is no substantial match, then the exemplary  
11 audio-content recognizer indicates, at 760, that the suspect clip 750 is not a  
12 substantial copy of the selected clip of 756. In other words, no piracy is detected  
13 if the final hash values of compared clips do not substantially match. At 764, this  
14 process ends.

15 However, if the result of such comparison is a substantial match, then the  
16 exemplary audio-content recognizer indicates, at 762, that the suspect clip 750 is a  
17 substantial copy of the selected clip of 756. In other words, piracy is detected if  
18 the final hash values of compared clips substantially match. At 764, this process  
19 ends.

## 20 Audio Content Categorization Methodological Implementation

21  
22 Fig. 8 illustrates an audio content categorization methodological  
23 implementation of the exemplary audio-content recognizer. At 816 of Fig. 8, the  
24 exemplary audio-content recognizer retrieves a hash value of a selected audio clip.  
25

1 More particularly, it retrieves the intermediate (i.e., categorization) hash value of  
2 such clip from a database of audio clips or some other source of such clips.

3 In dashed box 805, Fig. 8 also shows an alternative way of getting an  
4 intermediate hash value of the selected clip. This is by processing the clip using  
5 the audio-content identification method of Fig. 6 at block 812. The method 812  
6 calculates an intermediate (i.e., categorization) hash value of the selected clip. At  
7 814, the exemplary audio-content recognizer retrieves the calculated intermediate  
8 hash value of the selected audio content-base clip 810 from the audio-content  
9 identification method of block 812.

10 At 820, the exemplary audio-content recognizer uses the intermediate hash  
11 value of the selected clip to group such clip with others of similar (i.e., proximal)  
12 intermediate hash values. In other words, based upon the intermediate hash value  
13 of a given clip, the exemplary audio-content recognizer groups the given clip with  
14 other clips having similar intermediate hash values. Thus, the hash values of all  
15 clips in a given grouping are clustered together (i.e., proximal each other).  
16 Although these groupings are somewhat objectively determined, the subjective  
17 nature of the content of clips within a grouping will be similar to that of the  
18 content of others within the grouping.

19 The exemplary audio-content recognizer uses the categorization hash value  
20 of the selected clip to group such clip with others of similar (i.e., proximal)  
21 categorization hash values. In other words, based upon the categorization hash  
22 value of a given clip, the exemplary audio-content recognizer groups the given  
23 work with other works having similar categorization hash values. Thus, the hash  
24 values of all works in each grouping are clustered together (i.e., proximal each  
25 other). Although these groupings are primarily objectively determined, the

1 subjective nature of the content of works within a grouping will be similar to that  
2 of the content of others within the grouping.

3 The boundaries between groupings are determined manually or  
4 automatically. Manually, a person selects the boundary between groupings using  
5 the natural clustering seen after many clips have been categorized. Automatically,  
6 a system mathematically selects the boundary between groupings to be some point  
7 between (perhaps halfway) the centers of the groupings. Of course, other such  
8 techniques may be used to determine boundaries. These techniques may be fully  
9 automatic, fully manual, or some combination

10 At 822, the exemplary audio-content recognizer stores the categorization  
11 results in a database. At 824, the process ends.

### 12 **Exemplary Computing System and Environment**

13  
14 Fig. 9 illustrates an example of a suitable computing environment 900  
15 within which an exemplary audio recognizer, as described herein, may be  
16 implemented (either fully or partially). The computing environment 900 may be  
17 utilized in the computer and network architectures described herein.

18 The exemplary computing environment 900 is only one example of a  
19 computing environment and is not intended to suggest any limitation as to the  
20 scope of use or functionality of the computer and network architectures. Neither  
21 should the computing environment 900 be interpreted as having any dependency  
22 or requirement relating to any one or combination of components illustrated in the  
23 exemplary computing environment 900.

24 The exemplary audio recognizer may be implemented with numerous other  
25 general purpose or special purpose computing system environments or

1 configurations. Examples of well known computing systems, environments,  
2 and/or configurations that may be suitable for use include, but are not limited to,  
3 personal computers, server computers, thin clients, thick clients, hand-held or  
4 laptop devices, multiprocessor systems, microprocessor-based systems, set top  
5 boxes, programmable consumer electronics, network PCs, minicomputers,  
6 mainframe computers, distributed computing environments that include any of the  
7 above systems or devices, and the like.

8 Exemplary audio recognizer may be described in the general context of  
9 computer-executable instructions, such as program modules, being executed by a  
10 computer. Generally, program modules include routines, programs, objects,  
11 components, data structures, etc. that perform particular tasks or implement  
12 particular abstract data types. Exemplary audio recognizer may also be practiced  
13 in distributed computing environments where tasks are performed by remote  
14 processing devices that are linked through a communications network. In a  
15 distributed computing environment, program modules may be located in both local  
16 and remote computer storage media including memory storage devices.

17 The computing environment 900 includes a general-purpose computing  
18 device in the form of a computer 902. The components of computer 902 can  
19 include, by are not limited to, one or more processors or processing units 904, a  
20 system memory 906, and a system bus 908 that couples various system  
21 components including the processor 904 to the system memory 906.

22 The system bus 908 represents one or more of any of several types of bus  
23 structures, including a memory bus or memory controller, a peripheral bus, an  
24 accelerated graphics port, and a processor or local bus using any of a variety of  
25 bus architectures. By way of example, such architectures can include an Industry

1 Standard Architecture (ISA) bus, a Micro Channel Architecture (MCA) bus, an  
2 Enhanced ISA (EISA) bus, a Video Electronics Standards Association (VESA)  
3 local bus, and a Peripheral Component Interconnects (PCI) bus also known as a  
4 Mezzanine bus.

5 Computer 902 typically includes a variety of computer readable media.  
6 Such media can be any available media that is accessible by computer 902 and  
7 includes both volatile and non-volatile media, removable and non-removable  
8 media.

9 The system memory 906 includes computer readable media in the form of  
10 volatile memory, such as random access memory (RAM) 910, and/or non-volatile  
11 memory, such as read only memory (ROM) 912. A basic input/output system  
12 (BIOS) 914, containing the basic routines that help to transfer information  
13 between elements within computer 902, such as during start-up, is stored in ROM  
14 912. RAM 910 typically contains data and/or program modules that are  
15 immediately accessible to and/or presently operated on by the processing unit 904.

16 Computer 902 may also include other removable/non-removable,  
17 volatile/non-volatile computer storage media. By way of example, Fig. 9  
18 illustrates a hard disk drive 916 for reading from and writing to a non-removable,  
19 non-volatile magnetic media (not shown), a magnetic disk drive 918 for reading  
20 from and writing to a removable, non-volatile magnetic disk 920 (e.g., a "floppy  
21 disk"), and an optical disk drive 922 for reading from and/or writing to a  
22 removable, non-volatile optical disk 924 such as a CD-ROM, DVD-ROM, or other  
23 optical media. The hard disk drive 916, magnetic disk drive 918, and optical disk  
24 drive 922 are each connected to the system bus 908 by one or more data media  
25 interfaces 926. Alternatively, the hard disk drive 916, magnetic disk drive 918,

1 and optical disk drive 922 can be connected to the system bus 908 by one or more  
2 interfaces (not shown).

3 The disk drives and their associated computer-readable media provide non-  
4 volatile storage of computer readable instructions, data structures, program  
5 modules, and other data for computer 902. Although the example illustrates a hard  
6 disk 916, a removable magnetic disk 920, and a removable optical disk 924, it is to  
7 be appreciated that other types of computer readable media which can store data  
8 that is accessible by a computer, such as magnetic cassettes or other magnetic  
9 storage devices, flash memory cards, CD-ROM, digital versatile disks (DVD) or  
10 other optical storage, random access memories (RAM), read only memories  
11 (ROM), electrically erasable programmable read-only memory (EEPROM), and  
12 the like, can also be utilized to implement the exemplary computing system and  
13 environment.

14 Any number of program modules can be stored on the hard disk 916,  
15 magnetic disk 920, optical disk 924, ROM 912, and/or RAM 910, including by  
16 way of example, an operating system 926, one or more application programs 928,  
17 other program modules 930, and program data 932. Each of such operating  
18 system 926, one or more application programs 928, other program modules 930,  
19 and program data 932 (or some combination thereof) may include an embodiment  
20 of a digital audio signal hashing unit, a watermark encoder, transformer, a  
21 statistics estimator, an adaptive quantizer, an error-correction decoder, and a  
22 hasher.

23 A user can enter commands and information into computer 902 via input  
24 devices such as a keyboard 934 and a pointing device 936 (e.g., a "mouse").  
25 Other input devices 938 (not shown specifically) may include a microphone,

1 joystick, game pad, satellite dish, serial port, scanner, and/or the like. These and  
2 other input devices are connected to the processing unit 904 via input/output  
3 interfaces 940 that are coupled to the system bus 908, but may be connected by  
4 other interface and bus structures, such as a parallel port, game port, or a universal  
5 serial bus (USB).

6 A monitor 942 or other type of display device can also be connected to the  
7 system bus 908 via an interface, such as a video adapter 944. In addition to the  
8 monitor 942, other output peripheral devices can include components such as  
9 speakers (not shown) and a printer 946 which can be connected to computer 902  
10 via the input/output interfaces 940.

11 Computer 902 can operate in a networked environment using logical  
12 connections to one or more remote computers, such as a remote computing device  
13 948. By way of example, the remote computing device 948 can be a personal  
14 computer, portable computer, a server, a router, a network computer, a peer device  
15 or other common network node, and the like. The remote computing device 948 is  
16 illustrated as a portable computer that can include many or all of the elements and  
17 features described herein relative to computer 902.

18 Logical connections between computer 902 and the remote computer 948  
19 are depicted as a local area network (LAN) 950 and a general wide area network  
20 (WAN) 952. Such networking environments are commonplace in offices,  
21 enterprise-wide computer networks, intranets, and the Internet.

22 When implemented in a LAN networking environment, the computer 902 is  
23 connected to a local network 950 via a network interface or adapter 954. When  
24 implemented in a WAN networking environment, the computer 902 typically  
25 includes a modem 956 or other means for establishing communications over the

1 wide network 952. The modem 956, which can be internal or external to computer  
2 902, can be connected to the system bus 908 via the input/output interfaces 940 or  
3 other appropriate mechanisms. It is to be appreciated that the illustrated network  
4 connections are exemplary and that other means of establishing communication  
5 link(s) between the computers 902 and 948 can be employed.

6 In a networked environment, such as that illustrated with computing  
7 environment 900, program modules depicted relative to the computer 902, or  
8 portions thereof, may be stored in a remote memory storage device. By way of  
9 example, remote application programs 958 reside on a memory device of remote  
10 computer 948. For purposes of illustration, application programs and other  
11 executable program components such as the operating system are illustrated herein  
12 as discrete blocks, although it is recognized that such programs and components  
13 reside at various times in different storage components of the computing device  
14 902, and are executed by the data processor(s) of the computer.

### 15 **Computer-Executable Instructions**

16  
17 An implementation of an exemplary audio recognizer may be described in  
18 the general context of computer-executable instructions, such as program modules,  
19 executed by one or more computers or other devices. Generally, program modules  
20 include routines, programs, objects, components, data structures, etc. that perform  
21 particular tasks or implement particular abstract data types. Typically, the  
22 functionality of the program modules may be combined or distributed as desired in  
23 various embodiments.  
24  
25



## **Exemplary Operating Environment**

Fig. 9 illustrates an example of a suitable operating environment 900 in which an exemplary audio recognizer may be implemented. Specifically, the exemplary audio recognizer(s) described herein may be implemented (wholly or in part) by any program modules 928-930 and/or operating system 928 in Fig. 9 or a portion thereof.

The operating environment is only an example of a suitable operating environment and is not intended to suggest any limitation as to the scope or use of functionality of the exemplary audio recognizer(s) described herein. Other well known computing systems, environments, and/or configurations that are suitable for use include, but are not limited to, personal computers (PCs), server computers, hand-held or laptop devices, multiprocessor systems, microprocessor-based systems, programmable consumer electronics, wireless phones and equipments, general- and special-purpose appliances, application-specific integrated circuits (ASICs), network PCs, minicomputers, mainframe computers, distributed computing environments that include any of the above systems or devices, and the like.

## **Computer Readable Media**

An implementation of an exemplary audio recognizer may be stored on or transmitted across some form of computer readable media. Computer readable media can be any available media that can be accessed by a computer. By way of example, and not limitation, computer readable media may comprise “computer storage media” and “communications media.”

1 “Computer storage media” include volatile and non-volatile, removable and  
2 non-removable media implemented in any method or technology for storage of  
3 information such as computer readable instructions, data structures, program  
4 modules, or other data. Computer storage media includes, but is not limited to,  
5 RAM, ROM, EEPROM, flash memory or other memory technology, CD-ROM,  
6 digital versatile disks (DVD) or other optical storage, magnetic cassettes, magnetic  
7 tape, magnetic disk storage or other magnetic storage devices, or any other  
8 medium which can be used to store the desired information and which can be  
9 accessed by a computer.

10 “Communication media” typically embodies computer readable  
11 instructions, data structures, program modules, or other data in a modulated data  
12 signal, such as carrier wave or other transport mechanism. Communication media  
13 also includes any information delivery media.

14 The term “modulated data signal” means a signal that has one or more of its  
15 characteristics set or changed in such a manner as to encode information in the  
16 signal. By way of example, and not limitation, communication media includes  
17 wired media such as a wired network or direct-wired connection, and wireless  
18 media such as acoustic, RF, infrared, and other wireless media. Combinations of  
19 any of the above are also included within the scope of computer readable media.

## 20 **Conclusion**

21 Although the invention has been described in language specific towards  
22 digital audio signals, it is to be understood that the invention defined in the  
23 appended claims is not necessarily limited to digital audio signals. Rather, it may  
24  
25

1 apply to other digital signals (e.g., images, multimedia, video, film, data,  
2 information, text, etc.)

3 Although the invention has been described in language specific to structural  
4 features and/or methodological steps, it is to be understood that the invention  
5 defined in the appended claims is not necessarily limited to the specific features or  
6 steps described. Rather, the specific features and steps are disclosed as preferred  
7 forms of implementing the claimed invention.